

Tabla de contenido

1 Enunciado	1
2 Requisito de funcionamiento	1
3 Pasos a seguir	2
3.1.1 Calculadora	2
3.1.2 Estado de la calculadora	3
3.1.3 Acción Calculadora	3
3.1.4 Operación	4
4 Trabajo a realizar	4

1 Enunciado

Se pretende hacer una práctica completa de un pequeño proyecto relacionado con la calculadora.

De las entrevistas se han recogido los siguientes requisitos:

- La calculadora debe funcionar con números reales o números racionales
- Si en un estado cualquiera pasamos de un modo a otro se reiniciará a 0, no convertirá datos

2 Requisito de funcionamiento

El funcionamiento deseado de la calculadora es el que se expone a continuación:

Introducimos una operación de la forma **OP1 operador OP2**.

Posteriormente podremos dos acciones:

- ✓ presionar **=** y se visualizará el resultado.
- ✓ presionar **otro operador** y se visualizará el resultado seguido del nuevo operador, lista para la siguiente operación.

En función de que operemos en un modo u otro, la calculadora se adaptará gráficamente como se muestra a continuación:

- ✓ Si operamos en real aparecerá como división el símbolo **/**, y el símbolo de separación de decimales será el **.** (**punto**)
- ✓ Si operamos en racional, ese símbolo aparecerá como parte del número (separador de numerador/denominador), en este caso para dividir usaremos el símbolo **:**, y lógicamente desaparecerá el punto

Para simplificar sólo usaremos operaciones binarias (de dos operandos), según se expone a continuación

Suma (+), resta (-), multiplicación (*), división (/ :), Módulo (%)

C → Borrar

. → Separador de decimales en números racionales

= → Operador de obtener el resultado

3 Pasos a seguir

En la primera tarea debemos especificar el diagrama de clases. Para ello usaremos el modelador ArgoUML. En el diagrama de clases se pueden utilizar estereotipos. Los estereotipos es una herramienta disponible en UML, que me permite especificar un nuevo elemento a partir de otro existente. Para ello incluiremos <<estereotipo>>

P.E. si una clase implementa los métodos para decir si esOP o esNum o esClear Podremos crear un estereotipo que nos diga <<entrada>> es(). En este caso usamos el estereotipo para generalizar todos los métodos particulares de algo muy común.

Identificaremos los diagramas de clases indicando sólo los métodos y atributos principales, así por ejemplo la clase Calculadora que contiene la interfaz no hace falta en este análisis cual es el contenido de tallado de todos los atributos, sí que podemos usar un atributos estereotipado llamado <<interfazGrafica>> swing.

Una posible visión inicial de las clases nos podría quedar algo como lo que sigue

Calculadora, EstadoCalculadora, AccionCalculadora, Operacion

Una vez identificamos las clases vamos a ver lo que hace cada una, identificando los métodos que realiza.

3.1 Calculadora

Describiremos la interfaz gráfica.

Eventos que va a gestionar

- Click sobre algún botón

```
estado.cambiaEstado(c, tPantalla.getText(), accion, tipoCalculadora);  
setPantalla(accion.getCalculo());
```

- Cambiar el tipo de calculadora en el menú

```
setTitle("CALCULADORA TIPO");  
estado.inicializarEstado(accion);  
setPantalla(accion.getCalculo());  
//Cambiar botones según el tipo  
    brPto.setText(".");  
    brDiv.setText("/");  
tipoCalculadora=tipo leído;  
  
accion.setTipoCalculadora(tipoCalculadora);????
```

- Salir de la aplicación

3.2 Estado de la calculadora

En el caso de la clase **EstadoCalculadora** vamos a utilizar un diagrama de estados para describir su comportamiento. Se pide que hagamos el diagrama de estados teniendo en cuenta que sería posible identificar los siguientes estados

```
final int INICIAL =0;
final int OP1 =1;
final int OP1_REAL=2;
final int ESPERANDO_OP2 =3;
final int OP2 =4;
final int OP2_REAL =5;
final int RTDO =6;
final int ESPERANDO_DEN_OP1=7;
final int DEN_OP1 = 8;
final int ESPERANDO_DEN_OP2=9;
final int DEN_OP2 = 10;
```

Esta es una propuesta de estados para los casos de Real y Racional.

Estos estados cambian ante las siguientes entradas

```
private boolean esNum(char c){}
private boolean esSepRac(char c){}
private boolean esOpReal(char c){}
private boolean esOpRacional(char c){}
private boolean esIgual(char c){}
private boolean esPto(char c){}
private boolean esClear(char c){}
```

Donde , los valores respectivo son

- ✓ esNum (0,1,2,3,4,5,6,7,8,9)
- ✓ esSepRac (/)
- ✓ esOpReal(+ - * / %)
- ✓ esOpRacional (+ - * :)
- ✓ esIgual (=)
- ✓ esPto (.)
- ✓ esClear(C)

Cada estado tendrá un método que en función de la entrada cambiará el estado e invocará a la acción correspondiente.

3.3 AccionCalculadora

Y Estas son las acciones propuestas ante un determinado estado y una determinada entrada clase AccionCalculadora

- Actualiza (String) ➔ Actualiza la operación con la cadena de caracteres que le pasamos como entrada. El atributo cálculo contendrá la operación en un momento dado
- Calcula() ➔ Realiza el cálculo actual y retorna un string con el resultado
- actualizaCalculo(String c) (actualiza el cálculo añadiendo el sting)



3.4 Operación

Esta clase debe de realizar la operación a partir de un String que la contiene

- Cargar la operación a partir de un string
- Deberá sacar cada operador cada operador
- Obtener el operando
- Devolver el resultado

Hay que tener en cuenta que tenemos operandos reales y operandos racionales.

4 Trabajo a realizar

Se pide que en el proyecto realicemos una lista de requisitos que identifiquemos, un diagrama de clases y para la clase de estados diseñar un diagrama de estados .

5 Escribir el código fuente de cada clase.